Claims

1. (Currently Amended) A computerized method of creating test coverage for non-deterministic programs within a testing environment comprising:

in a computer, receiving a graph of edges and states representing a program under test, the states comprising at least one deterministic state controllable by the testing environment and at least one non-deterministic state uncontrollable by the testing environment;

creating a continuous cycle of edges through the graph that reaches each ~~edge~~ state in the graph at least once;

~~splitting the continuous cycle into discrete sequences that end at edges reaching non-deterministic states uncontrollable by the testing environment;~~

executing the program under test as a first execution of the program;

determining ~~untested program behavior as discrete sequences~~ untested states not reached by the first execution of the program;

calculating, for at least some deterministic states, a probability that during program execution, a path from the deterministic state will reach the at least one untested state;

calculating, for the at least some deterministic states, a number of edges between the at least one deterministic state and the untested state as the cost;

creating strategies through the graph to reach the untested state such that a next state with a lower cost and higher probability is preferred over a next state with higher cost and lower probability ~~that have a higher probability of reaching discrete sequences not reached by the program~~;

storing a representation of the created strategies in computer memory; and

in the computer, executing the program under test under test conditions using the stored created strategies such that ~~cause~~ the program execution ~~to have~~ has a higher probability than the first execution of the program to execute through states that correspond to the untested ~~program behavior~~ states.

2. (Canceled)

3. (Original) The method of claim 1 wherein the continuous cycle of edges is created from the graph input using a Chinese Postman tour algorithm.

4. (Original) The method of claim 1 wherein the graph states are received as a set of deterministic vertices and a set of non-deterministic vertices.

5. (Canceled)

6. (Canceled)

7. (Currently Amended) A computer system comprising:

memory and a central processing unit executing,

a compiler operationally able to compile ~~compiling~~ an executable specification into an abstract state machine ~~compiler~~;

a graphing program operationally able to create a continuous cycle touching all edges of the abstract state machine and operationally able to split ~~splitting~~ the continuous cycle into discrete sequences that end at non-deterministic states ~~program~~;

a program operationally able to execute the program in a test environment and determine untouched edges and states;

a calculating program operational to assign probabilities to deterministic states based on the probability that they will provide a path to an untouched discrete sequence, and using the assigned probabilities to calculate a strategy more likely to reach the untouched discrete sequences ~~program~~; ~~and~~

a program operationally able to ~~an executing~~ execute a test program and ~~verifying~~ verify that the test program executes states corresponding to those modeled by discrete sequences of the abstract state machine; ~~and determining untouched discrete sequences~~ and

a program operationally able to execute ~~executing~~ the test program according to the created strategies and ~~verifying~~ verify whether the program executes states corresponding to the untouched discrete sequences coverage ~~program~~.

8. (Original) The system of claim 7 wherein a continuous cycle is determined according to a Chinese Postman algorithm.

9. (Original) The system of claim 7 wherein discrete sequences comprise beginning states reachable from edges exiting non-deterministic states.

10. (Original) The system of claim 7 wherein an untouched discrete sequence is a state selectable from a program code executing at a remote computer.

11. (Original) The system of claim 7 wherein the abstract state machine comprises a graph of states and edges.

12. (Original) The system of claim 11 wherein the strategy calculation program receives the graph and an edge probability function as input.

13. (Previously Presented) The system of claim 7 wherein the strategy calculation program for creating strategies is executed to create created strategies; wherein untouched discrete sequences represent less than 10% of the discrete sequences and all untouched discrete sequences are touched when the test program is executed according to the created strategies.

14. (Previously Presented) The system of claim 7 wherein not all untouched discrete sequences are verified when the test program is executed according to the created strategies.

15. (Currently Amended) A tangible computer-readable medium having thereon computer-executable instructions comprising:

instructions stored on the computer-readable medium for creating a model of program behavior comprising an abstract state machine with edge transitions;

instructions stored on the computer-readable medium for verifying program behavior;

instructions stored on the computer-readable medium for splitting the model of program behavior into sequences of at least two edge transitions ending at non-deterministic behavior;

instructions stored on the computer-readable medium for determining strategies for the sequences of at least two edge transitions ending at non-deterministic behavior more likely to reach an identified program behavior,

wherein determined strategies are determined based on a ~~comparison~~ determining probability of edges exiting a deterministic state ~~representing program behavior~~ reaching a state representing the identified program behavior, and ~~a selected~~ selection of an edge having ~~has a~~ highest probability from among the probabilities determined of reaching ~~a~~ the state representing the identified program behavior; and

instructions stored on the computer-readable medium for causing a program to execute behavior corresponding to the strategies for the sequences of at least two edge transitions ending at non-deterministic behavior more likely to reach the identified program behavior.

16. (Canceled)

17. (Canceled)

18. (Original) The computer-readable medium of claim 15 wherein the non-deterministic behavior comprises communications with a remote computer.

19. (Canceled)

20. (Original) The computer-readable medium of claim 15 wherein the instructions for verifying program behavior cause the program to execute code that verifies that the program is in an expected model state.

21. (New) The method of claim 1 wherein calculating probability comprises calculating the probability that a nondeterministic state on a path from the deterministic state to the untested state will choose an edge that leads to the untested state.

22. (New) The method of claim 22 wherein the calculating the probability comprises determining the number of edges leaving the nondeterministic state as k, and calculating the probability as 1/k.

23. (New) The method of claim 1 further comprising walking backward from the untested state to a second deterministic state.

24. (New) The system of claim 7 further comprising, for a nondeterministic state with edges touching an untouched state, assigning a probability to each edge based on the likelihood that the edge will be selected.

25. (New) They system of claim 7 wherein assigning probabilities to states further comprises walking backward through the graph to assign the probabilities to the states.

26. (New) The system of claim 7 wherein a nondeterministic state is selected multiple times to increase the possibility that the nondeterministic state will select a desired edge.